TECHNISCHE UNIVERSITÄT DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

# Runtime Analysis of Binary PSO

Dirk Sudholt, Carsten Witt

No. CI-241/08

Technical Report ISSN 1433-3325 February 2008

Secretary of the SFB 531 · Technische Universität Dortmund · Dept. of Computer Science/LS 2 · 44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the Technische Universität Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

# Runtime Analysis of Binary PSO*

Dirk Sudholt
Fakultät für Informatik, LS 2
Technische Universität Dortmund
Dortmund, Germany

Carsten Witt
Fakultät für Informatik, LS 2
Technische Universität Dortmund
Dortmund, Germany

**Abstract**

We investigate the runtime of the Binary Particle Swarm Optimization (PSO) algorithm introduced by Kennedy and Eberhart (1997). The Binary PSO maintains a global best solution and a swarm of particles. Each particle consists of a current position, an own best position and a velocity vector used in a probabilistic process to update the particle's position. We present lower bounds for a broad class of implementations with swarms of polynomial size. To prove upper bounds, we transfer a fitness-level argument well-established for evolutionary algorithms (EAs) to PSO. This method is then applied to estimate the expected runtime on the class of unimodal functions. A simple variant of the Binary PSO is considered in more detail. The 1-PSO only maintains one particle, hence own best and global best solutions coincide. Despite its simplicity, the 1-PSO is surprisingly efficient. A detailed analysis for the function OneMax shows that the 1-PSO is competitive to EAs.

## 1 Introduction

The runtime analysis of randomized search heuristics is a growing area with many interesting results in the last decades. The analysis of evolutionary algorithms started with the investigation of simple evolutionary algorithms on simple example functions (see, e. g., Droste, Jansen and Wegener [2]). The theoretical results derived from such analyses then helped to develop methods to analyze more complex evolutionary algorithms on more complex problems. The runtime analysis of evolutionary algorithms can be called

*Supported by the Deutsche Forschungsgemeinschaft (SFB) as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

a success story since nowadays analyses are possible for many problems from combinatorial optimization. This includes, for example, maximum matchings [3], spanning tree problems [12, 11], matroid optimization [14] as well as the NP-hard partition problem [17].

In recent years, the first runtime analyses on swarm intelligence algorithms have appeared, following a similar approach as taken for the analysis of evolutionary algorithms. Such analyses are, in general, more difficult than for evolutionary algorithms as the probabilistic model underlying swarm algorithms may depend on a long history of past solutions. Regarding ant colony optimization (ACO), first runtime analyses have been presented independently by Gutjahr [4] and Neumann and Witt [13]. Neumann and Witt defined a simple ant algorithm called 1-ANT and analyzed its performance on the function OneMax. It turned out that the 1-ANT is very sensitive to the choice of the so-called evaporation factor that determines the amount of change in the probabilistic model. Similar results for other functions were presented by Doerr, Neumann, Sudholt and Witt [1]. On the other hand, variants of the Max-Min Ant System (MMAS) proved to be quite effective for simple functions (Gutjahr and Sebastiani [5] and Neumann, Sudholt and Witt [10]).

Particle swarm optimization (PSO) is another class of swarm algorithms that is mostly applied in continuous spaces. Originally developed by Kennedy and Eberhart [7], it has become a popular bio-inspired optimization principle in recent years. A comprehensive treatment is given in the book by Kennedy, Eberhart, and Shi [9]. A typical PSO algorithm maintains a swarm of particles where each particle corresponds to a solution of the problem at hand. Each particle moves through the search space according to a certain velocity. In every iteration the velocity of a particle is updated in the direction of its own best solution and the best individual in its neighborhood. This kind of behavior is motivated from social-psychology theory as it combines cognitive and social effects to determine the behavior of each particle.

Kennedy and Eberhart [8] presented a binary version of PSO, called *Binary PSO*. As in classical PSO, velocities are used to determine the next position of a particle. However, as each bit may only obtain discrete values 0 and 1, velocities are used in a stochastic solution construction process. More precise, the velocity value of a bit determines the probability to set this bit to 1 in the next solution construction. This closely relates to binary ACO algorithms like the 1-ANT or MMAS variants.

Our aim is to develop a theoretical understanding of the Binary PSO, in particular from the perspective of computational complexity. In the original formulation of Binary PSO, all velocities are restricted to an interval of

constant range. We prove in Section 2 that the effect on the performance is disastrous if $v_{\max}$ is fixed while the problem size grows. Instead, we present a formulation of the Binary PSO that is adjusted towards growing problem dimensions.

In Section 3 we present lower bounds on the runtime of Binary PSO. Section 4 shows how fitness-level arguments, a powerful tool for the analysis of evolutionary algorithms, can be used for the analysis of the Binary PSO using only the social component. We examplarily apply this technique to the class of unimodal functions. In Section 5, we consider a specific variant of Binary PSO in more detail. The 1-PSO works with a swarm consisting of only one particle. Despite its simplicity, the 1-PSO turns out to be surprisingly efficient. A thorough analysis on the function OneMax in Section 5 shows that the 1-PSO is competitive to evolutionary algorithms. We conclude in Section 6 with possible directions for future research.

## 2 The Binary PSO

We consider the Binary PSO algorithm by Kennedy and Eberhart [8] for the optimization of pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$. Generally, the Binary PSO algorithm maintains $\mu$ triples $(x^{(i)}, x^{*(i)}, v^{(i)})$, $1 \leq i \leq \mu$, denoted as particles. Each particle $i$ consists of its current position $x^{(i)} \in \{0,1\}^n$, its own best position $x^{*(i)} \in \{0,1\}^n$ and its velocity $v^{(i)} \in \mathbb{R}^n$. Note that the velocity is from a continuous domain. In PSO terminology, the three components of a particle are often called vectors. Using the language of optimization, we will refer to particle positions $x^{(i)}$, $x^{*(i)}$, and $x^*$ synonymously as solutions.

The movement for each particle is influenced by the best particle in its neighborhood. Hence, depending on the neighborhood structure, different particles may be guided by different good solutions. In this work, however, we only use the trivial neighborhood consisting of the whole swarm. This means that all particles are influenced by a single global best particle, denoted as $x^*$.

The velocities are updates as follows. The velocity vector is changed towards the particle's own best solution and towards the global best solution $x^*$. Using the language of social-psychology, the first component is often called cognitive component and the latter is often called social component. These impact of these two components is determined by so-called *learning factors* $c_1$ and $c_2$ representing parameters of the system. The factor $c_1$ is the learning factor for the cognitive component and $c_2$ is the one for the social

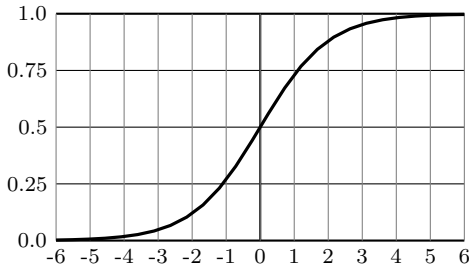


Figure 1: The sigmoid function $s(v) = \frac{1}{1+e^{-v}}$.

component. A common choice for the learning factors is to set $c_1 = c_2 = 2$.

We give a precise definition for the Binary PSO algorithm with a swarm size of $\mu$ and learning factors $c_1, c_2$. By lower indices we address the $n$ components of the three parts of the particle.

The algorithm starts with an initialization step (Step 1), where all velocities are set to all-zeros vectors and all solutions, including own best and global best solutions, are undefined, represented by the symbol $\perp$. The subsequent loop (Steps 2–5) chooses random scalars $r_1$ and $r_2$ anew in each iteration. These values are then used as weights for the cognitive and the social component, respectively. Using the language of evolutionary algorithms, we refer to iterations synonymously as generations.

In Step 3, the velocity is probabilistically translated into a new position for the particle, i. e., a new solution. As proposed in the original formulation, we use the sigmoid function

$$s(v) := \frac{1}{1 + e^{-v}}$$

shown in Figure 1. Hence positive velocity components bias the corresponding bit towards 1-values while negative velocities favor 0-values. At velocity $0^n$, each bit is completely random, hence the first created solution is uniformly distributed over $\{0, 1\}^n$.

Afterwards, the own best and global best solutions are exchanged if the newly constructed solution is better. Note that the selection is strict, i. e., a best solution is only exchanged in case the new solution has strictly larger fitness.

In Step 4 the Binary PSO performs some vector arithmetic to update the velocity vectors probabilistically in the direction to the particle's own best solution and the global best solution. To ensure convergence of the

heuristic, every velocity vector is bounded componentwise by minimum and maximum values, i. e., to an interval $[-v_{\max}, v_{\max}]$. This reflects the common choice of a maximum velocity as studied by Shi and Eberhart [15]. For practical purposes, often velocities in the interval $[-4, 4]$ are proposed. Since we will however conduct an asymptotic analysis, we allow the maximum velocity to grow with the problem dimension $n$ and confine the components to logarithmic values by letting $v_{\max} := \ln(n-1)$. We will justify this choice later.

**Algorithm 1** (Binary PSO)**.**

1. *Initialize velocities with* $0^n$ *and all solutions with* $\perp$*.*

2. *Choose* $r_1 \in U[0, c_1]$ *and* $r_2 \in U[0, c_2]$*.*

3. *For* $j := 1$ *to* $\mu$ *do*
   *For* $i := 1$ *to* $n$ *do*
   *Set* $x_i^{(j)} := 1$ *with probability* $s(v_i^{(j)})$*,*
   *otherwise set* $x_i^{(j)} := 0$*.*
   *If* $f(x^{(j)}) > f(x^{*(j)})$ *or* $x^{*(j)} = \perp$ *then* $x^{*(j)} := x^{(j)}$*.*
   *If* $f(x^{*(j)}) > f(x^*)$ *or* $x^* = \perp$ *then* $x^* := x^{*(j)}$*.*

4. *For* $j := 1$ *to* $\mu$ *do*
   *Set* $v^{(j)} := v^{(j)} + r_1(x^{*(j)} - x^{(j)}) + r_2(x^* - x^{(j)})$*.*
   *Restrict each component of* $v^{(j)}$ *to* $[-v_{\max}, v_{\max}]$*.*

5. *Goto 2.*

Throughout this work, we will deal with different implementations of the Binary PSO, differing in the swarm size $\mu$ and the learning factors $c_1$ and $c_2$. In particular, we deal with a remarkably simple yet effective algorithm, the so-called 1-PSO using just one particle. With just one particle, the own best solution and the global best solution coincide. Therefore, it makes sense to turn off the social component by setting $c_2 = 0$. The cognitive learning factor is set to the default value $c_1 = 2$. Note that the same algorithm is described by the choice $c_1 = 0$ and $c_2 = 2$. Dropping the upper index in the notation, the 1-PSO can be stated as follows.

**Algorithm 2** (1-PSO)**.**

1. *Initialize* $v = 0^n$ *and* $x^* = \perp$.
2. *Choose* $r \in U[0, 2]$.
3. *For* $i := 1$ *to* $n$ *do*
   *Set* $x_i := 1$ *with probability* $s(v_i)$,
   *otherwise set* $x_i := 0$.
   *If* $f(x) > f(x^*)$ *or* $x^* = \perp$ *then* $x^* := x$.
4. *Set* $v := v + r(x^* - x)$.
   *Restrict each component of* $v$ *to* $[-v_{\max}, v_{\max}]$.
5. *Goto 2.*

There are several good reasons to investigate the 1-PSO. One is that in the Binary PSO without social component, i. e., with $c_2 = 0$, all particles behave like independent instances of the 1-PSO. Moreover, by analyzing the 1-PSO we gain insight into the probabilistic model underlying the Binary PSO. This then helps to analyze more complex PSO variants. Finally, the investigation of the 1-PSO is interesting on its own as the 1-PSO turns out to be surprisingly effective.

## Lower Bound for Constant Velocity Range

The value $v_{\max}$ is often set to a constant value. This makes sense when dealing with problems of bounded size. However, one should be aware of the fact that for growing problem sizes a fixed value for $v_{\max}$ leads to an extreme decline in performance.

The following lower bound shows that if all velocities are restricted to constant values, then the Binary PSO is too close to random search and the algorithm fails badly, even given exponential time and a large number of global optima.

**Theorem 1.** *Consider the Binary PSO with arbitrary values for* $\mu$, $c_1$, *and* $c_2$, *where* $v_{\max}$ *is redefined to a constant value. Then there is a constant* $c = c(v_{\max})$ *such that the following holds. If* $f$ *contains at most* $2^{cn}$ *global optima, the probability that the Binary PSO finds a global optimum on* $f$ *within* $2^{cn}$ *constructed solutions is* $2^{-cn}$.

*Proof.* Choose $c$ such that $s(v_{\max}) = 2^{-3c}$ and note that $c$ is a positive constant if $v_{\max}$ is constant. We estimate the probability to construct any

specific solution $x$. Since the Binary PSO treats 0- and 1-bits symmetrically, we can w. l. o. g. assume that $x$ is the all-ones string $1^n$. Then even if all velocities are at $v_{\max}$, the probability to construct $x$ is still bounded by $(s(v_{\max}))^n = 2^{-3cn}$. By the union bound, the probability to construct any global optimum out of at most $2^{cn}$ ones is bounded by $2^{cn} \cdot 2^{-3cn} = 2^{-2cn}$. By the same argument, the probability that this happens at least once in $2^{cn}$ solution constructions is at most $2^{cn} \cdot 2^{-2cn} = 2^{-cn}$. □

For the common choice $v_{\max} := 4$, the constant $c$ is approximately 0.00873. As $2^{0.00873 \cdot n}$ is small for small $n$, the bad runtime behavior can only be observed if the problem size is large enough. This certainly isn't the case for $n = 100$ where $2^{cn} < 2$. However, for a problem size of $n = 10000$, the claimed bound has grown to $2^{cn} > 10^{26}$ and we would not expect to live long enough to see the Binary PSO find an optimum.

This analysis justifies our change in design, namely letting $v_{\max}$ scale with the problem size. In the following, we assume $v_{\max} = \ln(n-1)$. As $s(-v_{\max}) = 1/n$ and $s(v_{\max}) = 1 - 1/n$, the probability of setting a bit to 1 is always in the interval $[1/n, 1 - 1/n]$.

## 3 Lower Bound for Binary PSO

An important step towards runtime bounds for the Binary PSO is to understand the dynamics of the probabilistic model underlying PSO, that is, the behavior of the velocity vector. Consider a single bit that is set to 1 both in the own best and in the global best solution. Then, as long as these solutions are not exchanged, its velocity value $v$ is guided towards the upper bound $v_{\max}$. An important observation is that the velocity is only increased in case the bit is set to 0 in the next constructed solution. The probability that this happens is given by

$$1 - s(v) \;=\; 1 - \frac{1}{1+e^{-v}} \;=\; \frac{1}{1+e^{v}},$$

and we see that this probability decreases rapidly with growing $v$. Hence, the closer the velocity is to the bound $v_{\max}$, the harder it is to get closer. A symmetric argument holds for velocities that are guided towards $-v_{\max}$.

As long as the $v$-values are not too close to the velocity bounds $-v_{\max}$ and $v_{\max}$, the search of the Binary PSO is too random for it to find single optima with high probability. We can make this idea precise by the following, general lower bound, which holds for all practical choices of the learning factors $c_1$ and $c_2$ and a polynomial swarm size $\mu$.

**Theorem 2.** *Let $f$ be a function with a unique global optimum, let $\mu = \text{poly}(n)$ and let the sum $d := c_1 + c_2$ of the learning factors of the Binary PSO be $O(1)$. Then the expected number of generations of the Binary PSO on $f$ is $\Omega(n/\log n)$.*

*Proof.* W. l. o. g. the global optimum is $1^n$. Let $t := cn/\ln n$ for a small constant $c > 0$ which is chosen later. We show that the probability of not creating $1^n$ within $t$ generations is $1 - o(1)$, which implies the claim of the theorem.

We consider an arbitrary bit in an arbitrary particle. The event of creating a one at this bit is called success. Let a bit be called weak if its success probability has been at most $p := 1 - \frac{e \ln(\mu n)}{n}$ up to and including the current generation. Let a set of bits be called weak if it contains only weak bits. We will show that with probability $1 - 2^{-\Omega(n)}$ after $t$ generations of the 1-PSO, each particle contains still a weak subset of bits of size at least $n/e$. The probability of setting all bits of such a weak subset to 1 simultaneously is bounded from above by $p^{n/e} \leq 1/(\mu n)$ for each particle. Note that this event is necessary to create $1^n$ in a particle. Thus the probability of finding the optimum within $t$ generations creating $\mu$ new solutions each is still less than $t\mu/(\mu n) = O(1/\log n) = o(1)$, which will prove the theorem.

We still have to show that with probability $\Omega(1)$, after $t$ generations, there is a weak subset of size at least $n/e$ in each particle. One step can increase the velocity by at most $d$. Note that $p = 1 - O((\ln n)/n)$ since $\mu = \text{poly}(n)$. To reach success probability at least $p$, the current velocity must be between $s^{-1}(p) - d = \ln(p/(1-p)) - d$ and $s^{-1}(p)$ at least once. Pessimistically assuming the first value as current velocity, the probability of not increasing it in a single step is at least

$$\frac{1}{1 + e^{-\ln(p/(1-p))+d}} = 1 - \frac{e^d(1-p)}{p + e^d(1-p)} \geq 1 - 2e^d(1-p)$$

if $n$ is large enough for $p \geq 1/2$ to hold. The last expression equals $1 - (2e^d \ln n)/n$ by definition of $p$. Hence, along with $d = O(1)$ and again $\mu = \text{poly}(n)$, the probability of not increasing the velocity within $t$ steps is at least

$$\left(1 - \frac{2e^d \ln(\mu n)}{n}\right)^t = \left(1 - \frac{O(\ln n)}{n}\right)^{cn/\ln n} \geq 2e^{-1}$$

if $c$ is chosen small enough. This means that each bit in each particle independently has a probability of at least $2e^{-1}$ of being weak at generation $t$.

Using Chernoff bounds, the probability of not having a weak set of size at least $n/e$ in a specific particle is at most $e^{-\Omega(n)}$. As $\mu = \text{poly}(n)$, the probability that there exists a particle without weak subset at generation $t$ is still $\mu e^{-\Omega(n)} = e^{-\Omega(n)}$. □

# 4 Upper Bound for Binary PSO

In this section we derive an upper bound for a broad class of Binary PSO algorithms. Consider a Binary PSO where the cognitive component is turned off by setting $c_1 = 0$. Then each particle is driven only by its social behavior, that is, it tries to follow the leader of the swarm, the global best solution $x^*$. Note that this class of algorithms includes the 1-PSO. We will see that this setting allows the application of analysis tools known from evolutionary algorithms. In the following, we set $c_2$ to its default value 2, although results may be adapted for different constants.

The lower bound from Theorem 2 relied on the fact that a velocity that is guided towards $v_{\max}$ doesn't reach this value in short time and then the Binary PSO cannot find a single target efficiently. On the other hand, if we consider a longer period of time, the velocities may reach the bounds $-v_{\max}$ and $v_{\max}$, respectively. Then the Binary PSO samples within a promising region of the search space given by the global best solution.

In case a bit reaches the velocity bound corresponding to the global best solution, we say that the velocity has been "frozen" as the only chance to alter the velocity again is to have an improvement of the global best solution. The random time $F$ until a bit is frozen is called *freezing time*. The following lemma bounds this time by $O(n)$.

**Lemma 1.** *Consider the Binary PSO with $c_1 = 0$ and $c_2 = 2$. The expected freezing time for a single bit is bounded by $E(F) = O(n)$. Moreover, for $t \geq 8n(\ln n + 1)$, we have* $\text{Prob}(F \geq t) \leq 2e^{-t/(16n)}$.

*Proof.* Since the lemma must hold for arbitrary initial velocities, we need a worst-case initial value for the velocity at hand. Intuitively, $-v_{\max}$ should be such a value. More generally, definining $v^{(t)}$ to be the velocity at time $t$ if the initial velocity equals $v$, we would expect some kind of stochastic dominance according to $\text{Prob}(v^{(t)} \geq d) \geq \text{Prob}(w^{(t)} \geq d)$ for arbitrary $t$ and $d$ if $v \geq w$ holds. However, this is true only at a macroscopic level. Actually, if $w$ is only by a tiny amount larger than $v$, the dominance does not hold due to the above-mentioned slowdown of the process w. r. t. increasing values.

Fortunately, it can be shown that the dominance holds for $v \geq w + 2$. This means that a decrease of the initial value by at least 2 certainly slows

down the process. We therefore artificially extend the velocity scale by 2 and arrive at a simplified Markov process $v_t$, $t \geq 0$ called *v-process* on $[-v_{\max} - 2, v_{\max}]$ as follows. Initially, $v_0 := -v_{\max} - 2$. For $t \geq 0$, the random state $v_{t+1}$ is obtained as follows:

$$v_{t+1} := \begin{cases} \min\{v_t + r, v_{\max}\} & \text{with probability } \frac{1}{1+e^{v_t}} \\ v_t & \text{otherwise,} \end{cases}$$

where $r \in U[0, 2]$ is drawn independently. Due to the above-mentioned dominance, the $v$-process is a pessimistic model for the real velocities if we are looking for upper bounds on when to reach a given value.

The probability of increasing a value $v$ is $1 - s(v)$. Hence, the expected waiting time for an increase is $(1 - s(v))^{-1} = 1 + e^v$. By definition of $r$, each increase is bounded from below by 1 with probability at least $1/2$ (or $v_{\max}$ is reached anyway). Since this is independent of other steps, the expected waiting time for an increase by at least 1 (or up to $v_{\max}$) is bounded from above by $2 + 2e^v$. We obtain an upper bound on $E(F)$ if we sum up these waiting times for all integral values in $[\lfloor -v_{\max} - 2 \rfloor, \lceil v_{\max} \rceil]$. Since the waiting time is non-decreasing w. r. t. the $v$-value, this sum can be estimated by the corresponding integral. Hence, using the definition of $v_{\max}$,

$$\begin{aligned} E(F) \;&\leq \sum_{v=\lfloor -v_{\max}-2 \rfloor}^{\lceil v_{\max} \rceil} (2 + 2e^v) \;\leq\; \int_{\lfloor -v_{\max}-2 \rfloor}^{\lceil v_{\max} \rceil} (2 + 2e^v)\, \mathrm{d}v \\ &\leq\; 4(v_{\max} + 3 + e^{v_{\max}+1}) \;\leq\; 4 \ln n + 12 + 4en \;=\; O(n), \end{aligned}$$

which proves the first statement.

For the second statement, note that the probability of increasing a non-maximal $v$-value is always at least $1/n$. Since $t \geq 8n(\ln n + 1)$, we have $t/(4n) \geq 2v_{\max} + 2$. Hence, the following two events together are sufficient to reach $v_{\max}$ by time $t$:

- In $t$ steps there are at least $t/(2n)$ increases.
- The total amount of increase in $t/(2n)$ increases is at least $t/(4n)$.

We finish the considerations prematurely if $v_{\max}$ is reached with less increases or less total increase.

To bound the probability of failures, we use Chernoff and Hoeffding bounds. According to standard Chernoff bounds, the probability of less than $t/(2n)$ increases within $t$ trials is at most $e^{-t/(8n)}$. We can apply the Hoeffding bound from [6] for upper tails of random variables with bounded

range since the distributions of the considered random variables are symmetric. Hence, the increase in $t/(2n)$ steps is less than $t/(4n)$ with probability at most $e^{-t/(16n)}$. Altogether, $\text{Prob}(F \geq t) \leq 2e^{-t/(16n)}$. □

Due to the strict selection in the Binary PSO, $x^*$ is only exchanged in case a better solution is discovered. This means that after some time either the global best solution has improved or all velocities are frozen. In the latter case, since $v_{\max} = \ln(n-1)$, the probability to create a 1 for any bit is now either $s(-v_{\max}) = 1/n$ or $s(v_{\max}) = 1 - 1/n$. The distribution of constructed solutions equals the distribution of offspring of the (1+1) EA with $x^*$ as the current search point. For the sake of completeness, we give a definition of the (1+1) EA.

**Algorithm 3** ((1+1) EA)**.**

1. *Choose an initial solution $x^*$ uniformly at random.*
2. *For $i := 1$ to $n$ do*
   *Set $x_i := 1$ with probability $1 - 1/n$ if $x_i^* = 1$*
   *and with probability $1/n$ if $x_i^* = 0$.*
3. *If $f(x) \geq f(x^*)$ then $x^* := x$.*
4. *Goto 2.*

We also refer to the (1+1) EA* as the (1+1) EA with the condition in Step 3 replaced by $f(x) > f(x^*)$.

If for the 1-PSO all velocity values take their upper or lower bounds, the 1-PSO behaves like the (1+1) EA* until a solution with larger fitness is encountered. This similarity between PSO and EAs can be used to transfer a well-known method for the runtime analysis from EAs to PSO, the fitness-level method. We present this method, also called the method of $f$-based partitions (see [16]), in a restricted formulation. Let $f_1 < f_2 < \cdots < f_m$ be an enumeration of all fitness values and let $A_i$, $1 \leq i \leq m$, contain all solutions with fitness $f_i$. We also say that $A_i$ is the $i$-th fitness level. Note that the last fitness level $A_m$ contains only optimal solutions. Now, let $s_i$, $1 \leq i \leq m-1$, be a lower bound on the probability of the (1+1) EA (or, in this case equivalently, the (1+1) EA*) to create an offspring in $A_{i+1} \cup \cdots \cup A_m$, provided the current population belongs to $A_i$. The expected waiting time until such an offspring is created is at most $1/s_i$ and then the $i$-th fitness level is left for good. As every fitness level has to be

left at most once, the expected optimization time for the (1+1) EA and the (1+1) EA* is bounded above by

$$\sum_{i=1}^{m-1} \frac{1}{s_i}. \tag{1}$$

A similar bound holds for the Binary PSO using only the social component.

**Theorem 3.** *Let $A_i$ form the $i$-th fitness level of $f$ and let $s_i$ be the minimum probability for the (1+1) EA to leave $A_i$ towards $A_{i+1} \cup \cdots \cup A_m$. Consider a Binary PSO with $c_1 = 0$ and $c_2 = 2$. Let $\mu = \text{poly}(n)$, then the expected number of iterations to optimize $f$ is bounded from above by*

$$O\left(mn \log n + \frac{1}{\mu} \cdot \sum_{i=0}^{m-1} \frac{1}{s_i}\right).$$

Note that the right-hand sum is the upper bound obtained for the (1+1) EA and (1+1) EA* from (1). The factor $1/\mu$ reflects the fact that a large swarm may decrease the waiting time for an improvement. This behavior resembles a (1+$\lambda$) EA that creates $\lambda = \mu$ offspring in each generation. Note, however, that the number of $f$-evaluations is by a factor of $\mu$ larger than the number of iterations.

*Proof.* We only need to prove that the expected number of generations to increase the fitness from the $i$-th fitness level is at most $O(n \log n + 1/(\mu s_i))$.

We estimate the expected time until all bits in the swarm are frozen or an improvement happened anyway. Let $t := 32n(\ln n + \ln \mu)$. By Lemma 1, the probability that a single bit has not been frozen after $t$ generations is bounded by $2e^{-t/(16n)} = 2/n^2 \cdot 2/\mu^2$. By the union bound, the probability that all $\mu n$ bits in the swarm have not been frozen after $t$ iterations is at most $2/(\mu n)$. Considering independent phases of length $t$ each, the expected number of iterations until the swarm is frozen is at most $2t = O(n \log n)$.

Once all bits are frozen to the corresponding bounds of $x^*$, all particles behave equally until the next improvement. This implies that the Binary PSO performs $\mu$ trials in each generation to create a solution with higher fitness and the probability for a success in one trial is bounded below by $s_i$. The probability that the Binary PSO is not successful within a period of $1/s_i$ trials is bounded by

$$1 - (1 - s_i)^{1/s_i} \;\geq\; 1 - e^{-1} \;=\; \frac{e-1}{e}.$$

The expected number of periods is therefore bounded by $e/(e-1)$. The number of generations needed to have a period of $1/s_i$ trials equals $\lceil 1/(\mu s_i) \rceil$. Hence the expected number of generations to increase the fitness is bounded by

$$\frac{e}{e-1} \cdot \left( \frac{1}{\mu s_i} + 1 \right) \;=\; O\!\left( \frac{1}{\mu s_i} + 1 \right).$$

Adding the expected freezing time for the swarm yields the claimed bound $O(n \log n + 1/(\mu s_i))$ for fitness level $i$. □

The additive term $O(mn \log n)$ in the bound from Theorem 3 results from the (pessimistic) assumption that on all fitness levels, the Binary PSO has to wait until all velocities are frozen in order to find a better solution. Nevertheless, fitness-level arguments represent a powerful tool that can easily be applied to various problems. We examplarily present an application for unimodal functions.

A function $f$ is called unimodal if it has exactly one local optimum w. r. t. Hamming distance. Hence, if the global best solution $x^*$ is not the unique optimum, there is always at least one Hamming neighbor (a solution with Hamming distance 1 to $x^*$) with larger fitness. The probability for the (1+1) EA* to create a specific Hamming neighbor as offspring equals $1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$. We conclude $s_i \geq 1/(en)$ for every non-optimal fitness level. Theorem 3 yields the following bound.

**Corollary 1.** *Let $f$ be a unimodal function with $m$ different function values. Then the expected optimization time of the Binary PSO with $c_1 = 0$ and $c_2 = 2$ is bounded by*

$$O\!\left( mn \log n + \frac{1}{\mu} \cdot en \right) = O(mn \log n).$$

## 5 The 1-PSO on OneMax

Corollary 1 yields a bound $O(n^2 \log n)$ on the expected optimization time of the 1-PSO on OneMax, defined by

$$\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i.$$

Compared to the well-known bound $\Theta(n \log n)$ that holds for the (1+1) EA in this setting, this seems relatively rough. To improve the $O(n^2 \log n)$ bound, we have to show that it is not necessary for the 1-PSO to spend

$\Theta(n \log n)$ steps on each fitness level until all bits have been frozen to the velocity bounds of the global best solution.

In the following, we will improve the optimization time bound of the 1-PSO on OneMax to $O(n \log n)$. This implies that the 1-PSO has the same asymptotic upper runtime bound as the (1+1) EA. For the proof, we will basically show that $O(\log n)$ steps adjusting the velocity entries are enough on each fitness level for the 1-PSO to attain almost the same success probability as the (1+1) EA. Hence, a more careful inspection of the behavior of the velocities is required.

We reconsider the $v$-process as defined in the proof of Lemma 1. The random $v_t$, $t \geq 0$, gives us a random probability $P_t$ of setting the considered bit to 1 (called success) at time $t$. Its expectation $E(P_t) := \sum_p p \cdot \text{Prob}(P_t = p)$ equals the actual probability of a success at time $t$. However, it is important to study the distribution of $P_t$ and not only its expectation. The proof of Lemma 1 suggests that for $t = \Omega(n)$, $P_t$ is likely to be close to its maximum value $1 - 1/n$. Our following statement is more general.

**Lemma 2.** *Let $t \geq 16(\ln n + 2)$, $1 \leq i \leq t/96$ and $n$ be large enough. If $v_t$ is not capped by the upper bound $v_{\max}$ then*

$$\text{Prob}\left(P_t \geq 1 - \frac{96i}{t}\right) \geq 1 - e^{-i}.$$

*Proof.* Define $b(t,i) := s^{-1}(1 - 96i/t)$. We show the following claim: with probability at least $1 - e^{-i}$, it holds $v_t \geq b(t,i)$ or $v_t$ has reached $v_{\max}$ anyway; the latter case will pessimistically be ignored in the following. Since the success probability at value $b(t,i)$ is exactly $1 - 96i/t$, the claim implies the lemma.

Recall that the probability of increasing the $v$-value decreases monotonically with the $v$-value. It is therefore bounded from below by $96i/t$ before a $v$-value of at least $b(t,i)$ has been reached but increases with the distance of the current value from $b(t,i)$. A negative $v$-value, recall that $v_0 = -v_{\max} - 2$ (cf. Section 4, the $-2$ ensures a worst-case initial value), even leads to an increase with probability at least $1/2$. Altogether, a total increase by $v_{\max}+2+b(t,i)$ is sufficient to reach $b(t,i)$ by time $t$. We divide the progress to this boundary value into $\sqrt{t/(96i)} - 1$ phases with geometrically decreasing probabilities. During the $k$-th phase, $1 \leq k \leq \left\lceil \sqrt{t/(192i)} \right\rceil - 1$, the current success probability (i. e., the probability of *not* increasing the $v$-value) is within the interval $\left[1 - \frac{96(k+1)^2 i}{t},\ 1 - \frac{96k^2 i}{t}\right] \cap [1/2, 1]$. Since all

success probabilities are at least $1/2$ in the phases, it can be shown by taking the inverse sigmoid function that the length of an interval, expressed in $v$-scale, is at most $3\ln(k+1)$ if $n$ is not too small. We will show that with high probability, we spend at most $\frac{t\ln(k+1)}{4k^2}$ steps in a phase. If this holds for all phases, the total time spent in all phases is less than $\sum_{k=1}^{\infty}\frac{t\ln(k+1)}{4k^2} \le t/2$.

The times where the success probability of the $v$-process is less than $1/2$, i. e., where an increase happens with probability at least $1/2$, are treated separately. Arguing similarly as in the proof of Lemma 1, the following events together are sufficient to reach the desired $v$-value. Considering the second event, we use $t \ge 16(\ln n + 2)$, which implies $t/16 \ge v_{\max} + 2$. Note that Condition (3b) suffices to leave any considered interval of success probabilities.

1. In the first $t/2$ steps, there are at least $t/8$ increases, or a positive $v$-value is reached.
2. The total increase in $t/8$ increases is at least $t/16$, or a positive $v$-value is reached.
3. For $k = 1, \dots, \left\lceil\sqrt{\frac{t}{192i}}\right\rceil - 1$ it holds
   (a) If the current success probability is in the interval
   $$\left[1 - \frac{96(k+1)^2 i}{t},\ 1 - \frac{96k^2 i}{t}\right] \cap \left[\frac{1}{2}, 1\right]$$
   then $\frac{t\ln(k+1)}{4k^2}$ steps contains at least $12i\ln(k+1)$ increases.
   (b) The total increase in $12i\ln(k+1)$ increases is at least $3\ln(k+1)$.

We finish the considerations prematurely if the desired $v$-value is reached within less steps or less total increase.

According to Chernoff bounds, the failure probability for the first event is at most $e^{-t/32}$, which is less than $e^{-3i}$ since $i \le t/96$. According to Hoeffding bounds, the failure probability for the second event is at most $e^{-t/64} \le e^{-3i/2}$. Similarly, the failure probabilities for the third and fourth event are at most $e^{-3i\ln(k+1)}$ and $e^{-27i\ln(k+1)/8}$, in sum at most $2e^{-3i\ln(k+1)}$. Since $\sum_{k=1}^{\infty} e^{-3\ln(k+1)} \le 1/4$, the sum of the failure probabilities is at most $e^{-i}$ if $n$ is large enough. □

Instead of speaking of velocities at times $t$ or later, we introduce a handy notion.

**Definition 1.** *A random velocity is called* $t$-strong, $t \in \mathbb{N}_0$, *iff it stochastically dominates the v-process at time* $t$.

We also say that a bit is $t$-strong if this holds for its velocity. We summarize a simple fact: if a bit is currently $t$-strong, it will be $t+t'$-strong after another $t'$ steps provided the $x^*$-entry for this bit is 1 during these times.

Using Lemma 2, we know enough about the distribution of a $t$-strong bit to show the following claim. We consider this bit in the $v$-process where the $x^*$-entry never changes.

**Lemma 3.** *The expected time for a success at a* $t$*-strong bit,* $384 \ln n \leq t \leq n^2$*, is bounded by* $1 + O(1/t + 1/n)$.

*Proof.* We ignore the upper bound $1-1/n$ on success probabilities and allow a probability to become arbitrarily close to 1. Since the expected time for a success at success probability $1 - 1/n$ equals $1/(1 - 1/n) = 1 + O(1/n)$, the asymptotic upper bound of the lemma is not affected.

We consider a random variable $\tilde{P}$ with support $\{1-96i/t \mid 1 \leq i \leq t/192\}$ and distribution

$$\begin{aligned} \text{Prob}(\tilde{P} = 1 - 96/t) &= 1 - e^{-1}, \\ \text{Prob}(\tilde{P} = 1 - 96i/t) &= e^{-i+1} - e^{-i} \text{ for } 2 \leq i \leq t/192, \\ \text{and } \text{Prob}(\tilde{P} = 1/n) &= e^{-t/192}. \end{aligned}$$

Obviously, all probabilities sum up to 1. Since $t \geq 384 \ln n$, the last assignment implies $\text{Prob}(\tilde{P} = 1/n) \leq 1/n^2$.

Using Lemma 2, is follows that the success probability of the $t$-strong bit stochastically dominates $\tilde{P}$. It is therefore enough to bound the expected time for a success according to $\tilde{P}$ from above. Given that $\tilde{P}$ has the value $p$, the waiting time for a success follows a geometric distribution with expectation $1/p$. We can bound the reciprocals of the single success probabilities according to $1/(1 - 96i/t) \leq 1 + 97i/t$ if $n$ is not too small. By the law of total probability, the unconditional expected waiting time is at most

$$\begin{aligned} &(1 - e^{-1}) \cdot \left(1 + \frac{97}{t}\right) + \sum_{i=2}^{t/192} \left(e^{-i+1} - e^{-i}\right) \cdot \left(1 + \frac{97i}{t}\right) + \frac{1}{n^2} \cdot n \\ &\leq \left(1 + \frac{97}{t}\right) + \sum_{i=2}^{t/192} \left(e^{-i+1} - e^{-i}\right) \frac{97i}{t} + O\left(\frac{1}{t} + \frac{1}{n}\right) \\ &\leq \left(1 + \frac{97}{t}\right) + \sum_{i=2}^{t/192} e^{-i+1} \frac{97i}{t} + O\left(\frac{1}{n}\right) = 1 + O\left(\frac{1}{t} + \frac{1}{n}\right). \end{aligned}$$

□

Using a very similar calculation, we will show the following lower bound on the expected success probability at time $t$. Note that this lower bound is basically the reciprocal of the expected success time we have just derived. However, since we are dealing with random probabilities, the following lemma does not imply the preceding one.

**Lemma 4.** *The expected success probability of a $t$-strong bit, $384 \ln n \leq t \leq n^2$, is bounded by $1 - O(1/t)$.*

*Proof.* We reuse the random variable $\tilde{P}$ as defined in the proof of Lemma 3. Recall that the real success probability of the $t$-strong bit stochastically dominates $\tilde{P}$. Therefore, using the law of total probability, the expected success probability is at least

$$\begin{aligned}
&(1 - e^{-1}) \cdot \left(1 - \frac{96}{t}\right) + \sum_{i=2}^{t/192} \left(e^{-i+1} - e^{-i}\right) \cdot \left(1 - \frac{96i}{t}\right) \\
&\geq\ (1 - e^{-1}) \cdot \left(1 - \frac{96}{t}\right) + (e^{-1} - e^{-t/192}) - \sum_{i=2}^{t/192} e^{-i+1} \cdot \frac{96i}{t} \\
&\geq\ (1 - e^{-t/192}) \left(1 - \frac{96}{t}\right) - O\left(\frac{1}{t}\right) \ =\ 1 - O\left(\frac{1}{t}\right),
\end{aligned}$$

which completes the proof. □

In the following analysis, we will consider $k$ random velocities of 1-bits gained while optimizing OneMax. Freshly gained 1-bits tend to have a weaker velocity than older ones. Sorting the bits from weak to strong, this will be reflected by the following layering.

**Definition 2.** *The values $v_1, \ldots, v_k$ of $k$ velocities form an $m$-*layer*, $m \in \mathbb{N}$, iff $v_j$, $1 \leq j \leq k$, is $jm$-strong. A set of $k$ velocities forms an $m$-layer iff it can be arranged as an $m$-layer.*

We also say that bits form an $m$-layer if their velocities form such a layer. Again it is helpful to summarize a simple fact: given that $i$ bits form an $m$-layer, we can consider any other bit and wait for it to become $m$-strong. If $x^*$ remains fixed for these $i+1$ bits, they form an $m$-layer after at most $m$ steps.

For the following theorem, we will consider layers where the $j$-th bit is basically $\Theta(j \ln n)$-strong. Defining that a set of bits is successful if all, independently processed, have successes simultaneously, we show the following lemma. Note that we again consider these bits in the $v$-process and assume $x^*$ to be fixed to 1 for these bits.

**Lemma 5.** *Let $k \leq n$ independent bits form a $(384 \ln n)$-layer. Then the expected time until all have a success simultaneously is bounded by $O(1)$.*

*Proof.* Due to the independence, it suffices to multiply the expected success times for the single bits. According to our assumption, the $j$-th bit, $1 \leq j \leq k$, is $384j \ln n$-strong. By Lemma 3, its expected success time is bounded by $1 + \frac{\kappa}{384j \ln n} + \frac{\kappa}{n}$ for some large constant $\kappa$. Taking the product over all $j$, we obtain

$$\prod_{j=1}^{k} \left(1 + \frac{\kappa}{384j \ln n} + \frac{\kappa}{n}\right) \leq e^{\sum_{j=1}^{n} \frac{\kappa}{384j \ln n} + \frac{\kappa}{n}},$$

which is $O(1)$ since $\sum_{j=1}^{n} 1/j = O(\ln n)$. □

Now we can state the improved bound for OneMax.

**Theorem 4.** *The expected optimization time of the 1-PSO on* OneMax *is $O(n \log n)$.*

*Proof.* The basic proof idea is to keep track of the velocities of the newly gained 1-bits after improvements of the best-so-far solution $x^*$. We wait on average $O(\log n)$ steps after an improvement and show that after that, the probability of improving is at least in the same order as for the (1+1) EA.

A difficulty with these arguments is that 1-bits in $x^*$ may be set to 0 if the best-so-far solution is exchanged. We call this a reset of a bit. Resets may disturb the velocity increase on 1-bits as strong 1-bits may be replaced by weaker 1-bits.

In order to simplify the argumentation, we first describe an analysis for an idealized setting and then argue how to extend the arguments to the real setting. Assume in the following that the 1-PSO does not accept resets of 1-bits, i. e., an improvement of the OneMax-value is only accepted in case all 1-bits are set to 1 in the new best-so-far solution.

We now divide a run of the 1-PSO into phases. Phase 0 only contains the initialization step. Phase $i$ for $1 \leq i \leq n$ starts with the end of the previous phase and it ends when the following two conditions are met:

1. The best-so-far OneMax-value is at least $i$.
2. At least $i$ 1-bits form an $m$-layer for $m := 384 \ln n$.

Note that the second condition will be fulfilled throughout the run as all 1-bits are maintained forever in our idealized setting and hence their velocities are monotone over time.

We claim that the expected time spent in Phase $i$ is bounded above by $O(\ln n + n/(n-i))$ for each $1 \leq i \leq n$. Note that phases may be empty. Moreover, when finishing Phase $n$ the global optimum has been found. Hence, the expected time to find a global optimum is bounded by

$$\begin{aligned}\sum_{i=1}^{n} O\left(\ln n + \frac{n}{n-i}\right) &= O(n \log n) + O(n) \cdot \sum_{i=1}^{n} \frac{1}{i} \\ &= O(n \log n).\end{aligned}$$

Consider the 1-PSO at the time it enters Phase $i$. As Phase $i-1$ has been completed, $i-1$ 1-bits form an $m$-layer. According to Lemma 5, all these bits are set to 1 simultaneously after an expected number of $O(1)$ steps. Independently of these bits, the 1-PSO turns each 0-bit into a 1-bit with probability at least $1/n$, hence the probability of turning at least one 0-bit into 1 is at least $\Omega((n-i)/n)$. The expected waiting time for this event is $O(n/(n-i))$. Due to the independence, we can multiply expectations. Altogether the expected time until constructing a solution with OneMax-value at least $i$ has been bounded from above by $O(n/(n-i))$.

Once the best-so-far OneMax-value has increased to at least $i$, the velocities on $i$ 1-bits are monotone increasing. Since currently $i-1$ bits form an $m$-layer, the $i$ 1-bits by definition form an $m$-layer after at most $m = O(\ln n)$ steps. Together, the claimed bound $O(\ln n + n/(n-i))$ follows for the expected time in Phase $i$. This also finishes the analysis for the idealized setting without resets.

A reset of a bit can destroy the velocity layers as a strong 1-bit with might be exchanged by a weak 1-bit. In the worst case, such a new 1-bit is only 0-strong. If an improvement resets $d$ bits, an $m$-layer of $i$ bits may shrink to an $m$-layer of $i-d$ 1-bits. By an amortized analysis, we wait for the velocities to recover so that we end up with an $m$-layer of $i$ bits again.

Consider an improvement in a setting where $k$ bits form an $m$-layer. A $t$-strong bit is reset with probability at most $O(1/t)$ according to Lemma 4. The expected number of bits among these $k$ layered bits reset during this improvement is therefore bounded from above by

$$\sum_{j=1}^{k} \frac{O(1)}{384j \ln n} = O(1).$$

Hence, an improvement prolongs the time spent in the current phase in expectancy by $O(\ln n)$. Note that we can repeat the argumentation if another

improvement occurs in the meantime since we only consider reset probabilities for all bits in a layer. As we can only have $n$ improvements, we obtain an additional term $O(n \log n)$ in our runtime bound, which proves the time bound $O(n \log n)$ for the real setting. □

# 6 Conclusions and Future Work

We have considered the runtime behavior of the Binary PSO algorithm by Kennedy and Eberhart. Thereby, we adapted the choice of the maximum velocity $v_{\max}$ to growing problem sizes and justified why this adaptation is necessary when dealing with large problem sizes. For the resulting Binary PSO we have proved a lower bound $\Omega(n/\log n)$ on the expected number of generations for any function where the global optimum is unique. This bound holds for almost any choice of the swarm size and the learning factors $c_1$ and $c_2$ for the cognitive and the social component of PSO.

We also assessed the impact of these two PSO components. The Binary PSO using only the social component behaves similar to a $(1+\lambda)$-evolutionary algorithm with population size 1 and an offspring population of size $\lambda$ as all particles are guided by one global best particle. Due to this similarity, we were able to transfer a fitness level argument from the analysis of evolutionary algorithms (EAs) to PSO. The upper bounds derived from this method do not differ much from bounds known for EAs. An exemplary application to the class of all unimodal functions showed that the Binary PSO is effective on these functions.

On the other hand, if only the cognitive component is used in the Binary PSO, all particles behave independently and can be seen as many instances of the simple 1-PSO, a Binary PSO using just one particle. Our results on the 1-PSO may be applied in such a setting. Despite its simplicity, the 1-PSO is surprisingly efficient. A detailed analysis on the function OneMax revealed the runtime bound $O(n \log n)$ for the 1-PSO and hence the same upper bound as known for the (1+1) EA.

Future work should focus on the runtime of the Binary PSO when cognitive and social effects melt together. Then, the Binary PSO performs an update both towards the own best and towards the global best solution. This can also be seen as an update towards a recombined solution for own best and global best. For a bit where own best and global best differ, this yields a tendency for the velocity towards value 0, that is, for assigning the bit randomly. This resembles a genetic algorithm with uniform crossover. However, as velocities are not necessarily guided towards $-v_{\max}$ or $v_{\max}$,

this may prevent the velocity vector from freezing. Thus, it is difficult to apply fitness level arguments. In order to obtain upper bounds in such a setting, different arguments need to be developed.

## References


[1] B. Doerr, F. Neumann, D. Sudholt, and C. Witt. On the runtime analysis of the 1-ANT ACO algorithm. In *Proc. of GECCO '07*, pages 33–40. ACM, 2007.

[2] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theor. Comput. Sci.*, 276:51–81, 2002.

[3] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proc. of STACS '03*, volume 2607 of *LNCS*, pages 415–426, 2003.

[4] W. J. Gutjahr. First steps to the runtime complexity analysis of Ant Colony Optimization. *Computers and Operations Research*, 2008. To appear.

[5] W. J. Gutjahr and G. Sebastiani. Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability*, 2008. (to appear).

[6] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58(301):13–30, 1963.

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.

[8] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proc. of the World Multiconference on Systemics, Cybernetics and Informatics (WMSCI)*, pages 4104–4109, 1997.

[9] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.

[10] F. Neumann, D. Sudholt, and C. Witt. Comparing variants of MMAS ACO algorithms on pseudo-boolean functions. In *Proc. of SLS 2007*, volume 4638 of *LNCS*, pages 61–75, 2007.

[11] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[12] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theor. Comput. Sci.*, 378(1):32–40, 2007.

[13] F. Neumann and C. Witt. Runtime analysis of a simple Ant Colony Optimization algorithm. In *Proc. of ISAAC '06*, volume 4288 of *LNCS*, pages 618–627. Springer, 2006. Extended version to appear in Algorithmica.

[14] J. Reichel and M. Skutella. Evolutionary algorithms and matroid optimization problems. In *GECCO '07*, pages 947–954, 2007.

[15] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Proc. of the Seventh Annual Conference on Evolutionary Programming*, pages 591–600, 1998.

[16] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R. Sarker, X. Yao, and M. Mohammadian, editors, *Evolutionary Optimization*, pages 349–369. Kluwer, 2002.

[17] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS '05*, volume 3404 of *LNCS*, pages 44–56, 2005.